environment 535, if the method is interpreted, runtime system 546 may obtain the method from runtime environment 535 in the form of a sequence of bytecodes 530, which may be directly executed by interpreter 544. If, on the other hand, the method which is invoked is a compiled method which has not been compiled, runtime system 546 also obtains the method from runtime environment 535 in the form of a sequence of bytecodes 530, then may go on to activate compiler 542. Compiler 542 then generates machine instructions from bytecodes 530, and the resulting machine-language instructions may be executed directly by processor 402 of FIG. 4. In general, the machine-language instructions are discarded when virtual machine 540 terminates. The operation of virtual machines or, more particularly, Java™ virtual machines, is described in more detail in The Java™ Virtual Machine Specification by Tim Lindholm and Frank Yellin (ISBN 0-201-63452-X), which is incorporated herein by reference.

checked. SXL 12/15/04

[0039] Java classes (and interfaces) are dynamically loaded, linked and initialized. Loading is the process of the system finding the binary form of the class (e.g., the class file) and constructing from the binary form a Class object to represent the class. The Class class is a class for storing or representing the structures of classes. Linking is the process of taking a binary form of the class and combining it into the runtime state of the system so that it may be executed. Initialization of a class includes executing the class' static initializers and initializers for static fields declared in the class.

[0040] Each Java class has a constant pool associated with it. The constant pool is stored in the Java class file and serves a function similar to symbol tables. Typically, each entry in the constant pool is indexed by a number starting with one and ending with the number of entries in the constant pool. A method for a class accesses entries in the constant pool by the index and a method for one class may not access a constant pool for another class.

[0041] In addition to the constant pool storing literal constants, the constant pool stores classes, methods, fields, and interfaces symbolically. By storing these entries symbolically it is meant that the name identifying the entry is stored, not the physical address. In other words, if a class A has a field F, both the names of A and F (along